A POS-based Preordering Approach for English-to-Arabic Statistical Machine Translation

Mohamed Seghir Hadj Ameur, Ahmed Guessoum, and Farid Meziane

NLP, Machine Learning and Applications (TALAA) Group
Laboratory for Research in Artificial Intelligence (LRIA)
Department of Computer Science, University of Science and Technology Houari Boumediene (USTHB)
Bab-Ezzouar, Algiers, Algeria
{mhadjameur,aguessoum}@usthb.dz, f.meziane@salford.ac.uk

Abstract. In this work, we present a POS-based preordering approach that tackles both long- and short-distance reordering phenomena. Syntactic unlexicalized reordering rules are automatically extracted from a parallel corpus using only word alignment and a source-side language tagging. The reordering rules are used in a deterministic manner; this prevents the decoding speed from being bottlenecked in the reordering procedure. A new approach for both rule filtering and rule application is used to ensure a fast and efficient reordering. The tests performed on the IWSLT2016 English-to-Arabic evaluation benchmark show a noticeable increase in the overall Blue Score for our system over the baseline PSMT system.

Keywords: Machine Translation, Arabic NLP, Preordering, Reordering Rules, Statistical Translation

1 Introduction

When translating between two languages that are noticeably different in terms of their grammatical structures, the task of producing a high-quality translation in a correct word order becomes a serious challenge. Finding a better way to model these grammatical transformations or what is known as reordering phenomena, was and still is a long standing problem which receives a great deal of attention from the machine translation community. The classic Phrase-based Statistical Machine Translation System (PSMT) [1–3] has two means for word reordering: it can either learn the whole bi-phrase as an entry in the phrase table (generally a bi-phrase length does not exceed a certain limit) or via the distortion model which allows limited phrase movements (reorderings) in the output, but with a certain penalty. These means cannot address reorderings that involve a long distance jump or what is known as long-distance reordering. This problem represents a well-known limitation for the standard PSMT system, hence the need to provide a more sophisticated model to solve it.

Syntactic reordering in machine translation is a research area that aims to find more efficient solutions so as to handle both short- and long-distance reordering problems. One common way to perform reordering as a standalone process is known as preordering. Preordering is a preprocessing step that precedes the PSMT phase; its goal is to minimize the syntactic gap between languages and make their grammatical construction as close as possible. Preordering is commonly used to address the long-distance reordering problems, it has the advantage of being easy to use and independent from the used translation system.

Figure 1 gives an example of short- and long-distance word reordering phenomena performed on an aligned English-to-Arabic sentence pair. Both the English and Arabic texts are written from left-to-right to keep the alignment order consistent. In the first example (a) the word "announced" appears at the end of the English sentence, aligned to the Arabic word "أعلن" by means of the 6^{th} alignment link. Preordering will attempt to swap the position of the first and the 6^{th} alignment links, which moves the word "announced" to the beginning of the English sentence to match the Arabic sentence order as shown in the second example (b). Since these two links are separated by a large margin we call this reordering a long-distance word reordering. Short reordering cases such as the one involving the second and the third links are considered as performing short-distance reordering.

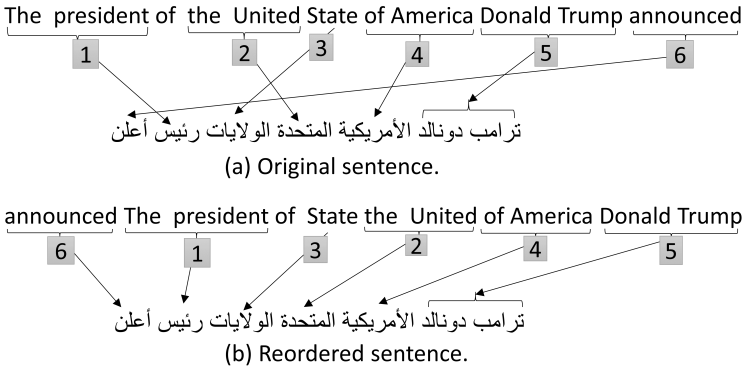

Fig. 1: An example illustrating the process of word reordering performed on an English-to-Arabic word-aligned sentence pair. (The English and Arabic texts are written from left-to-right to keep the alignment order consistent)

The goal of word preordering is to perform a reordering process on the English side of the corpus prior to the translation phase. This is done by finding and applying a set of syntactic rules which helps tweak the grammatical construction of the English-side language making it as close as possible to the Arabic-side language structure, which can also be seen as the task of minimizing the alignment links, as shown in Fig. 1 (b).

In this work, we propose a preordering method that can efficiently handle both long- and short- distance word reorderings. The reordering rules are learned

automatically from a parallel corpus using word alignment and a basic part-of-speech source language tagging. The test results showed a noticeable improvement over the baseline PSMT system which proves the consistency and adequacy of our proposal. The remainder of this paper is organized as follows: the next section gives an overview of the state-of-the-art preordering methods. The used PSMT baseline is then presented in Sect. 3. Section 4 presents our proposed reordering system and explains the details of each of its components. In Sect. 5, we present and discuss the tests we have done and the results we have obtained. Finally, in the last Section, we conclude our work and highlight some possible future improvements.

2 Related Work

Preordering methods can be classified into two main categories: the deterministic approaches which provide the decoder with only one optimal reordering; and the non-deterministic methods which feed the decoder with multiple candidate sentences in the form of a weighted lattice, and it is then up to the decoder to find the best choice among them.

In terms of deterministic methods, one of the earlier works was done by Xia et al. [4], whose system deals with the task of French-to-English machine translation. They automatically extracted syntactic rules (which they called rewrite patterns) from a bilingual corpus, using syntactic parsers of the source and target languages along with word alignment. They reported a 10% relative improvement in the Blue Score. Habash [5] proposed a preordering method for Arabic-to-English translation. He used word alignment and a source dependency parse tree to automatically extract syntactic reordering rules. The extracted rules were used to reorder the Arabic training and testing data. He investigated various alignment strategies and parsing representations and provided a comparative analysis of the different combinations of the investigated strategies. Genzel [6] defined the reordering task as a dependency parse tree transformation, in which the goal is to find the best children order for each internal node that has more than two children. He proposed a number of metrics for rule quality estimation which allows the filtering and selection of higher quality reordering rules. His proposal was tested on the task of translation from English to various other languages. In a similar fashion, Yang et al. [7] performed the reordering task on a dependency parse tree by reordering the children of each internal node. They handled the position of each node as its rank making the reordering a ranking problem in which the task is to find a certain function f that determines the best rank of each child. Then, the children get sorted according to their ranks. For the task of translation from Chinese to Japanese, Sudoh et al. [8] used a learning-to-rank model based on a pairwise classification method to predict the target Japanese word order. In the same spirit, Jehl et al. [9] proposed a feature-based reordering model for English-to-Japanese and English-to-Korean translation. Their model predicts whether a pair of sibling nodes on the source-side of the parse tree needs to be swapped. Based on the node swapping probabil-

ities, a global branch-and-bound search is applied to find the best ordering of the children. Fuji et al. [10] proposed a global reordering model that captures language-specific sentence structure directly from non-annotated corpora and use it to boost the performance of conventional syntactic reordering system.

In terms of non-deterministic methods, Zhang et al. [11] presented a pre-ordering strategy for Chinese-to-English translation using chunk-based syntactic rules. They used a source-reordering lattice instead of a single best reordering, and a reordering source language model as an additional feature to score each path in the lattice. Elming [12] presented a preordering approach for English-to-Danish translation. His proposed approach automatically learns probabilistic rules from a parallel corpus. The reordered sentences are fed via a lattice to the SMT decoder. He reported an absolute improvement in the translation quality of 1.1% in Blue Score.

Despite the existing work on word preordering, to the best of our knowledge, no strategy has appeared to give ideal reordering results, hence the continuing efforts to improve them. This work aims to introduce a new, efficient way for both rule identification and application, along with a method for estimating rules usefulness in the reordering process.

3 PSMT Baseline System

Given a source sentence f that we want to translate into a target sentence e. The phrase-based statistical machine translation [1–3] finds the best translation $\hat{e}$ from the space of all possible translations of f.

$$\hat{e} = argmax_e = p(e|f) \tag{1}$$

This can be decomposed using the noisy channel decomposition [13] into a translation model $p(f|e)$ and a language model $p(e)$.

$$\hat{e} = argmax_e = (p(e) * p(f|e)) \tag{2}$$

The translation model ensures the accuracy of the translation between the source and the target languages, and the language model ensures the fluency of the generated target sentences.

A more common generalization is the log-linear model [13] which, instead of splitting the problem into a translation and a language model, it enables the incorporation of arbitrary components (or features), with the assumption that these components are independent from each other,

$$\hat{e} = argmax_e = \left(\frac{exp^{\sum_1^M \alpha_m \varphi_m(f,e)}}{\sum_{e'} exp^{\sum_1^M \alpha_m \varphi_m(f,e')}}\right) \tag{3}$$

where M is the number of components, $\varphi_m(f,e)$ is the m^{th} component and α_m is its corresponding weight in the log-linear model.

The denominator $\sum_{e'} exp^{\sum_1^M \alpha_m \varphi_m(f,e')}$ being constant for all possible translations e', it can be omitted at decoding.

$$\hat{e} = argmax_e = exp^{\sum_1^M \alpha_m \varphi_m(f,e)} \quad (4)$$

Since the log-linear components are supposed to be independent, they can be trained separately. After training each component, an optimization technique such as the Minimum Error Rate Training (MERT) [14] can be used to find the optimal component weights. Our baseline log-linear model includes the following components:

- Phrase translation model
- Language model
- Distance-based reordering model
- Word penalty

4 Preordering System

Our proposed preordering system automatically learns syntactic reordering rules and uses them to change the grammatical structure of the English source-side sentences making them as close as possible to the target Arabic one.

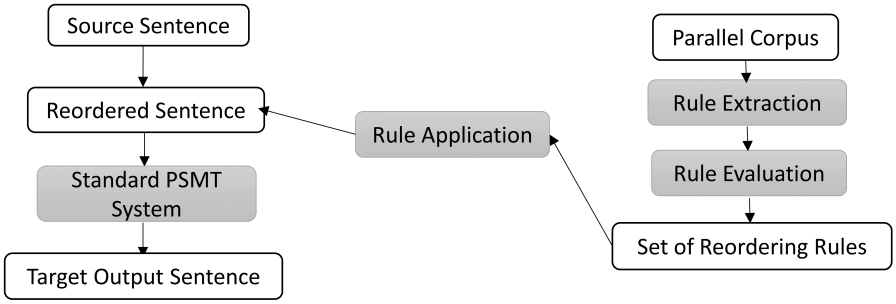

Fig. 2: Architecture of the preordering framework

Figure 2 shows the architecture of our proposed preordering system. There are two main steps: first, a set of reordering rules will be extracted from a parallel corpus, then each rule will be evaluated using a specific rule evaluation mechanism. The second step applies the selected rules to reorder both the training and testing data prior to their exploitation in the PSMT.

4.1 Reordering Rules Definition

In this work, the reordering rules are composed of part-of-speech tags only; as such all the used rules are unlexicalized. Two tagsets are considered: the English Penn Treebank (PTB) tagset (48 tags) [15], and the English Universal (Univ) tagset (17 tags) [16].

Using high-level tags (more general tag classes) will result in more general rules, while using more specific tags will allow the rules to capture more accurate contextual information albeit with a low generalization ability. The intuition behind using these two different tagsets is to investigate in a practical way the impact of the tags fineness on the reordering performance.

Our reordering rules are composed of three parts: the condition, the reordering, and an optional context. A rule condition may have more than one possible reordering, each reordering having its own specific context.

Table 1: An example of reordering rules

Rule Number	Rule Condition	Rule Action (Reordering)	Rule Context
1	DT NNP NNPS	2, 0, 1	(IN, IN)
2	IN DT NNP NNPS	3, 0, 1, 2	(NN, IN)
3	DT NN IN ... VBD	10, 0, 1, ..., 9	(Non, Non)

Table 1 shows an example of reordering rules with PTB part-of-speech tags. The first column presents the condition part of the rule and the second column shows all its corresponding reorderings. The context is presented as a pair (previous tag, next tag) which need to appear before and after the condition part of the rule. For example, for the first rule, the sequence of tags "DT NNP NNPS" need to be present in the sentence. Additionally, the left and right contextual tags "**IN** DT NNP NNPS **IN**" also need to appear before and after the condition. In such case, the reordering "2, 0, 1" can be applied to reorder the tags producing a new order "NNPS DT NNP".

4.2 Reordering Rules Extraction

The reordering rules are extracted using only word alignment and a tagged source text of the parallel corpus. Figure 3 shows the overall process of rules extraction.

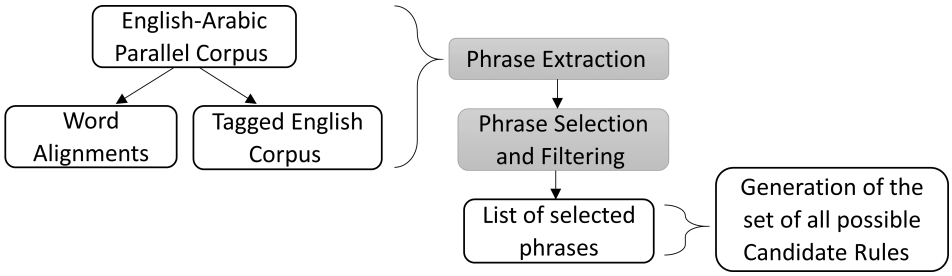

Fig. 3: Rules extraction mechanism

First, the bi-phrases are extracted using word alignment with a tagged source text, the phrases are then filtered by imposing some restrictions, and finally, the candidate rules are formed from the selected phrases.

Figure 4 shows an example of word alignment with a tagged source text. The Univ and PTB tags are presented for each word in the English source sentence. In the first step, phrase extraction is done using the standard phrase extraction

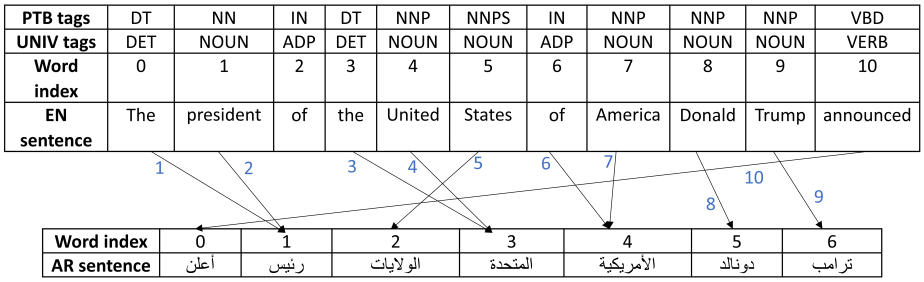

Fig. 4: Word alignment with tagged source-side, in which both the English and Arabic texts are written from left-to-right to keep the alignment order consistent

algorithm described by Koehn [13], which uses word alignment to extract bi-phrases from a parallel corpus. From our previous example of Fig. 4, the phrase extraction algorithm extracts a total of 25 bi-phrases, some of which are shown in Table 2.

Table 2: Some extracted bi-phrases from the example given in Fig. 4 using the standard phrase extraction algorithm described in [13]

English Phrase	Arabic Phrase
Donald Trump	دونالد ترامب
the United States	الولايات المتحدة
announced	أعلن

Having a set of English-Arabic bi-phrases, we select the pair of bi-phrases that contain a crossing. For instance, in Fig. 4 the phrases denoted by the links 4 and 5 cross each other (by abuse of language, since the links cross each other); thus swapping them will minimize the number of crossing links; which makes the English sentence structure more similar to the Arabic one. We will be using this kind of crossings of bi-phrases in-order to form our syntactic rules.

Given two bi-phrases $p_1 = (s_1, t_1)$ and $p_2 = (s_2, t_2)$ where s_i and t_i are phrases from the source and target sentences, respectively, the two bi-phrases p_1 and p_2 are considered valid to form a syntactic rule if they satisfy the following conditions:

1. If s_1 precedes s_2 in the source sentence, then t_2 must precede t_1 in the target sentence. In other words, the two bi-phrases must cross each other.
2. The two bi-phrases must be consecutive in both the source and the target sentences. In other words, s_2 must follow s_1 and t_2 must follow t_1.

For example in Fig. 4, the two bi-phrases denoted by the links 5 and 4 respect these two conditions. The two bi-phrases 5 and 3 do not respect the second condition (they are not consecutive in the English text).

The set of selected phrases are then used to generate unlexicalized syntactic rules; the left and right tags that precede and follow the two phrases are used as context. Table 1 shows some valid rules that can be extracted from Fig. 4.

4.3 Reordering Rules Evaluation

The extracted rules are not always useful for reordering. In fact, most of them are very specific, which makes their coverage quite limited. Another issue resides in the errors introduced by the automatic word alignment which increases the rate of incorrect rules.

To tackle these problems, a number of metrics that estimate rules quality have been proposed. The metric that is most used is the Crossing Score (CS) [6] which determines the quality of a rule based on the decrease in crossing alignment links after its application.

In practice, the quality of a rule is tested on the whole training corpus by applying the rule to each of its sentences and evaluating the change in the number of crossing alignments. This should give a solid estimation of the rule quality.

Applying this kind of metric directly will be computationally expensive since each rule is generally evaluated separately. Another issue is to perform the reordering task when given a set of rules. This involves finding all the applicable rules and determining the best order for their application. To this end, we build an index that accelerates both rules lookup and rules application.

Index Construction We build an index, which is a compact Trie [17]. To reduce rule lookup time, this index will be used for the tasks of rules evaluation and application.

Formally, given a set of rules R, with their conditions driven from a set of tags G, such that $|R| = n$ and $|G| = m$. Each rule r in R, is a tuple (c, a, x), where c is the condition, a is the action and x is the rule context.

We construct a compact Trie T on R which has the following characteristics:

- A root node and n leaves (since each rule condition c ends at a leaf node which contains its corresponding action a and context x).
- For each internal node, its descendants have the same prefix (the same condition).
- Two branches leaving the same node can't start with the same prefix.

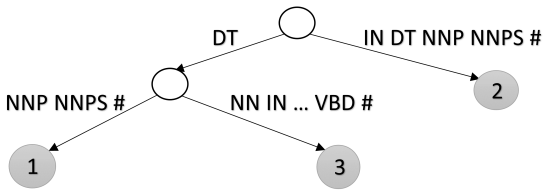

Fig. 5: Syntactic rules indexing via a compact Trie

Figure 5 shows a compact Trie constructed over the tag-sequences of the previous set of rules from Table 1. The leaf nodes are numbered according to their corresponding rules and the labels are printed on the edges. A special end tag # is added to ensure that each sequence terminates at the level of a leaf node. The reordering and context for each rule are kept in the leaf node that corresponds to it.

Efficient Search for Applicable Rules The task of finding all the applicable rules for a given sentence is very challenging given the number of rules and the variations in their corresponding part-of-speech tags. Algorithm 8 presents an easy and efficient way to identify all the applicable rules for a given sentence using a compact Trie representation for all the reordering rules. In-order to find

Algorithm 1: Algorithm *FindAll* that finds all the applicable rules for a given sentence

Input : T: a Trie constructed over a set of rules R.
$S_t = t_1, t_2, t_k$: the part-of-speech tags of the English sentence S where k is the length of S.
Output: *candidate_rules*: a list that contains all the applicable rules for S.

```
1 Function FindAll(R, T, S_t):
2     foreach suffix s_t in S_t starting at postion i do
3         rules_i = finds all applicable rules for s_t in T;
4         foreach rule r in rules_i do
5             candidate_rules.add ((r, i))
6         end
7     end
8 return candidate_rules;
```

all the applicable rules for a sentence S, Algorithm 8 finds all the applicable rules for each suffix (each position) in the tagged source sentence S_t; this is done by traversing the Trie starting from the root node and following the path led by each suffix in s_t. The rules found for each suffix are accumulated in *candidate_rules* and returned when the algorithm terminates. All the applicable rules for S can be found in $O(k^2)$ time, where k is the length of S.

Having the set of applicable rules for a given sentence, we need to determine the best one among them. This is done by sorting the obtained candidate rules according to their condition part to ensure that the rules concerning long-distance reorderings are applied first.

Rule Quality Evaluation As mentioned in Sect. 4.3, since the majority of rules are not useful for reordering purposes, a good method for rule quality estimation is needed. Algorithm 2 scores the rules which can be applied to a given sentence using the CS metric.

Algorithm 2 starts by identifying the best applicable rule for a given sentence S as described in Sect. 4.3. A close list is then used to prevent the rules from being reused in the same position. The best rule is then applied to reorder the word-aligned sentence, and the number of crossing alignments is then estimated using the CS metric. The score of the applied rule is then updated based on the CS difference [1]. This process is repeated for several iterations as indicated by the *max_iterations* variable.

[1] *FindCS* is a simple method that finds the number of crossing alignments (CS) for a given aligned sentence.

Algorithm 2: Algorithm *UpdateSent* that updates the evaluation score for each syntactic rule

Input : R: a set of rules.
T: a Trie constructed over R.
$S_t = t_1, t_2, t_k$: S part-of-speech tags for the sentence S.
S_a: alignment points for the sentence S and its target translation.
close: a close list.
Output: Updates the scores for each applicable rule in T for the sentence S.

Function `UpdateSent`(R, T, S_t, S_a, *close*)**:**
  $original_{CS} = findCS(S_a)$
  while $i < max\_iterations$ **do**
    $candidate\_rules = FindAll(R, T, S_t)$ that are not present in *close*;
    r_{best} = find the best rule in *candidate_rules*;
    insert r_{best} in *close*;
    S'_a = reorder S_a using r_{best};
    $new_{CS} = findCS(S'_a)$;
    r_{best}.usage+=1;
    if $new_{CS} < original_{CS}$ **then**
      r_{best}.positive+=1;
    end
    else if $new_{CS} > original_{CS}$ **then**
      r_{best}.negative+=1;
    end
    else
      r_{best}.neutral+=1;
    end
    $i = i + 1$;
  end

After scoring all the rules, we estimate the usefulness of a given rule r by taking the ratio of the number of time the rule gave a positive impact on the reordering task and the total number of its application:

$$usefulness(r) = \frac{positive(r)}{usage(r)} \tag{5}$$

In case the rule usefulness surpasses a certain threshold, it will be considered useful and selected for reordering. Applying this equation on the whole corpus will select a subset of useful reordering rules (since not all the rules can be applied in all possible contexts). This process of rules usefulness estimation is repeated for several iterations until the subset of the useful rules stabilizes, which indicates that a convergence point has been reached.

Figure 6 shows the variation of the number of selected rules when estimating the rules' usefulness.

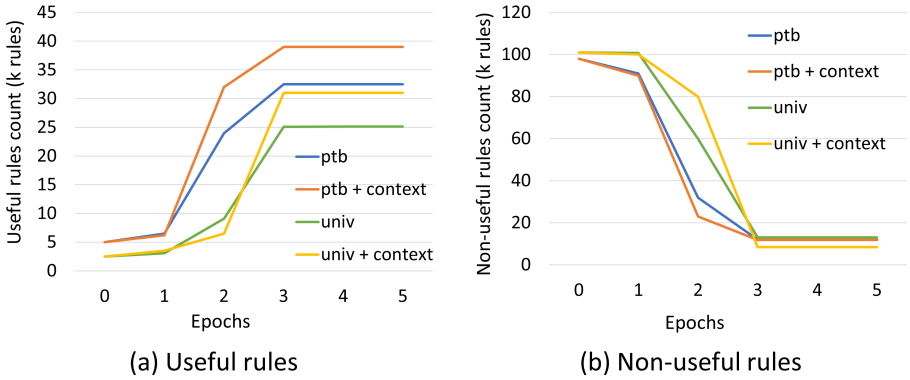

Fig. 6: Rules count variation when applying the process of rules filtering

The number of useful rules (Fig. 6 (a)) increases with the number of epochs and at the same time, the count of non-useful rules (Fig. 6 (b)) decreases until a convergence is achieved.

5 Experiments

To test our approach, we have used the English-Arabic parallel corpus provided by the IWSLT2016 [2] evaluation campaign which offers a complete testing framework which includes: training, development, and evaluation data. Our results have been obtained on the IWSLT 2010, 2011, 2012, 2013 and 2014 test sets. Tagging is done using the Stanford English Log-linear Part-Of-Speech Tagger [18]. The Univ part-of-speech tags are obtained by converting the PTB tagset using a simple tag mapping method [3].

5.1 Preprocessing

For the Arabic language, our preprocessing includes: diacritic sign removal, Arabic character normalization and word segmentation by means of the AMIRA toolkit [19] using the default tokenization scheme in which conjunctions, prepositions, determinants, suffixes and future markers are all individually separated. For the English side, only word tokenization is performed using the Python NLTK toolkit [4]. We have also added a number $<nbr>$ and a link classes $<url>$ to all numbers and links found in the parallel corpus respectively. Sentence length has been limited to 40 words; "bad" sentence pairs, i.e. whose length difference exceeds a certain threshold were also removed.

Table 3 shows some statistics about the resulting data from the preprocessing step.

[2] `http://workshop2016.iwslt.org/59.php`

[3] The conversion table can be found in the following link `http://universaldependencies.org/tagset-conversion/en-penn-uposf.html`

[4] `http://www.nltk.org/`

Table 3: Statistics about the training corpus

	English	Arabic
Sentences	110 549	110 549
Words	1692394	1910968
Unique words	26574	37539

5.2 Evaluation of Translation Quality

We have investigated the use of two tagsets and the presence/absence of part-of-speech contexts. This leads to four systems:

1. Reordering with Univ tagset without context.
2. Reordering with PTB tagset without context.
3. Reordering with Univ tagset with context.
4. Reordering with PTB tagset with context.

All our systems have been tested using the Moses PSMT framework [20]. We have used a 6-gram language model instead of the default tri-gram model to ensure a better language modeling for the segmented Arabic language. The rest of the parameters are kept unchanged. Our test results have been reported using the Blue Score Metric [21].

Table 4: Blue Score results for the PSMT baseline and the MSE-bidirectional reordering model

Test set	PSMT-Baseline	PSMT-MSE-Bi
IWSLT2010	17.24	17.33 (+0.09)
IWSLT2011	17.28	17.54 (+0.26)
IWSLT2012	19.30	19.48 (+0.18)
IWSLT2013	18.67	18.64 (-0.03)
IWSLT2014	16.16	16.82 (+0.66)

Table 4 shows the Blue Score results obtained by the Moses baseline with and without its default MSE-bidirectional reordering model [22]. The values in parentheses indicate the gain in Blue score with respect to the PSMT baseline system. A slight increase in Blue Score is obtained when the default Moses reordering model was turned on.

Table 5: Blue Scores using the PTB and the Univ part-of-speech tags without including the context

Test set	Base-UNIV	Base-PTB
IWSLT2010	17.03 (-0.21)	17.52 (+0.28)
IWSLT2011	17.62 (+0.34)	18.18 (+0.90)
IWSLT2012	19.80 (+0.50)	19.95 (+0.65)
IWSLT2013	18.73 (+0.06)	19.11 (+0.44)
IWSLT2014	17.22 (+1.06)	17.51 (+1.35)

Table 5 shows the reordering results obtained when using the PTB tags and the Univ tags without including the context. The obtained results when using the PTB was slightly better than the one obtained with the Univ tags. The

Table 6: Blue Scores using the PTB tags and the Univ tags when including the context

Test set	CONTEXT-UNIV	CONTEXT-PTB
IWSLT2010	17.72 (+0.48)	17.71 (+0.47)
IWSLT2011	18.31 (+1.03)	18.34 (+1.06)
IWSLT2012	19.90 (+0.6)	20.09 (+0.79)
IWSLT2013	19.13 (+0.46)	19.24 (+0.57)
IWSLT2014	17.61 (+1.45)	17.58 (+1.42)

maximum gain in Blue Score was 1.35 point compared to the Baseline PSMT system.

Table 6 shows the reordering results obtained using the PTB tags and the Univ tags when the context is included. The obtained results for the two tagsets were very similar with a maximum increase of about 1.5 in the Blue Score over the PSMT baseline. These results prove the importance of using the context to enhance the accuracy of the syntactic rules. Indeed, the more specific the rules, the better. Another thing to note is the effect of the POS-tag fineness: we can see that the use of PTB tags yields better results than with Univ tags, especially when no context is used. This suggests that more tag fineness will lead to a more accurate reordering.

5.3 Evaluation of Alignment Ambiguity

We have also used the Normalize Crossing Links Score (NCS) [6] to measure the quality of the different investigated reordering systems. The NCS metric formula is the following:

$$NCS = \frac{C}{S} \tag{6}$$

where C is the number of crossing links in the aligned corpus and S is the number of words in the source text of the corpus.

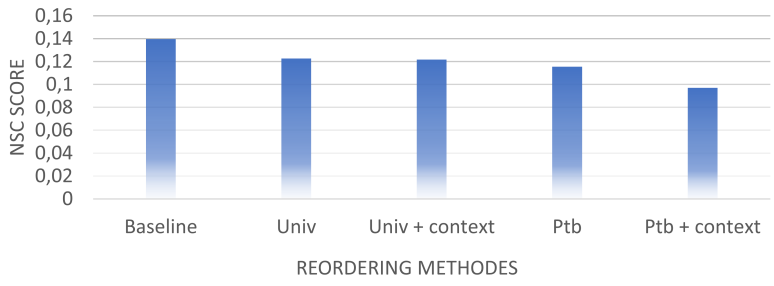

Fig. 7: The NCS scores for the different reordering methods

For this formula the smallest the NCS score, the better. An ideal score will be zero, which means that the corpus is completely monotonic [5]. The NCS

[5] We mean by a monotonic corpus, a corpus in which the alignment does not contain any crossing links.

scores are shown in Fig. 7 for the different reordering methods. We recall that the smallest the score, the better. The results indicate that using the PTB tagset with the contextual information produce less ambiguous alignments, hence better translation results.

6 Conclusion

We have introduced a general method for word preordering in which reordering rules are extracted from a parallel corpus using only word alignments and basic part-of-speech tagging. Rule quality is estimated using the CS metric, which allows the selection of only the best applicable rules. Our proposal has been evaluated in terms of translation quality using the Blue Score, and the change in alignment ambiguity has been investigated using the NCS metric. We have found out that using the PTB tags yield a more noticeable improvement over the baseline PSMT system; this suggests that the higher the tag fineness the better the effect of the part-of-speech preordering methods.

As a future work, we plan to explore a similar approach using tree structures (dependency and constituency trees). We also plan to examine the coupling of both preordering and post-ordering strategies in the same framework and check whether that yields to further improvements in the overall translation performance.

References

1. Brown, P.F., Cocke, J., Della-Pietra, S.A., Della-Pietra, V.J., Jelinek, F., Lafferty, J.D., Mercer, R.L., Rossin, P.: A statistical approach to machine translation. Computational Linguistics **16**(2) (1990) 76–85
2. Zens, R., Och, F.J., Ney, H.: Phrase-based statistical machine translation. In: Annual Conference on Artificial Intelligence, Springer (2002) 18–32
3. Och, F.J., Ney, H.: Discriminative training and maximum entropy models for statistical machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics (2002) 295–302
4. Xia, F., McCord, M.: Improving a statistical mt system with automatically learned rewrite patterns. In: Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics (2004) 508
5. Habash, N.: Syntactic preprocessing for statistical machine translation. Proceedings of the 11th MT Summit (2007) 10
6. Genzel, D.: Automatically learning source-side reordering rules for large scale machine translation. In: Proceedings of the 23rd international conference on computational linguistics, Association for Computational Linguistics (2010) 376–384
7. Yang, N., Li, M., Zhang, D., Yu, N.: A ranking-based approach to word reordering for statistical machine translation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Association for Computational Linguistics (2012) 912–920

8. Sudoh, K., Nagata, M.: Chinese-to-japanese patent machine translation based on syntactic pre-ordering for wat 2016. In: Proceedings of the 3rd Workshop on Asian Translation (WAT2016). (2016) 211–215
9. Jehl, L., Gispert, A., Hopkins, M., Byrne, W.: Source-side preordering for translation using logistic regression and depth-first branch-and-bound search. (2014)
10. Fuji, M., Utiyama, M., Sumita, E., Matsumoto, Y.: Global pre-ordering for improving sublanguage translation. WAT 2016 (2016) 84
11. Zhang, Y., Zens, R., Ney, H.: Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In: Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation, Association for Computational Linguistics (2007) 1–8
12. Elming, J.: Syntactic reordering integrated with phrase-based smt. In: Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation, Association for Computational Linguistics (2008) 46–54
13. Koehn, P.: Statistical Machine Translation. 1st edn. Cambridge University Press, New York, NY, USA (2010)
14. Och, F.J.: Minimum error rate training in statistical machine translation. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, Association for Computational Linguistics (2003) 160–167
15. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The penn treebank. Computational linguistics **19**(2) (1993) 313–330
16. Petrov, S., Das, D., McDonald, R.: A universal part-of-speech tagset. arXiv preprint arXiv:1104.2086 (2011)
17. De La Briandais, R.: File searching using variable length keys. In: Papers presented at the the March 3-5, 1959, western joint computer conference, ACM (1959) 295–298
18. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, Association for Computational Linguistics (2003) 173–180
19. Diab, M.: Second generation amira tools for arabic processing: Fast and robust tokenization, pos tagging, and base phrase chunking. In: 2nd International Conference on Arabic Language Resources and Tools. Volume 110. (2009)
20. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, Association for Computational Linguistics (2007) 177–180
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics (2002) 311–318
22. Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M., Talbot, D., White, M.: Edinburgh system description for the 2005 iwslt speech translation evaluation. In: IWSLT. (2005) 68–75